Clasificación de imágenes con métodos clásicos y redes neuronales

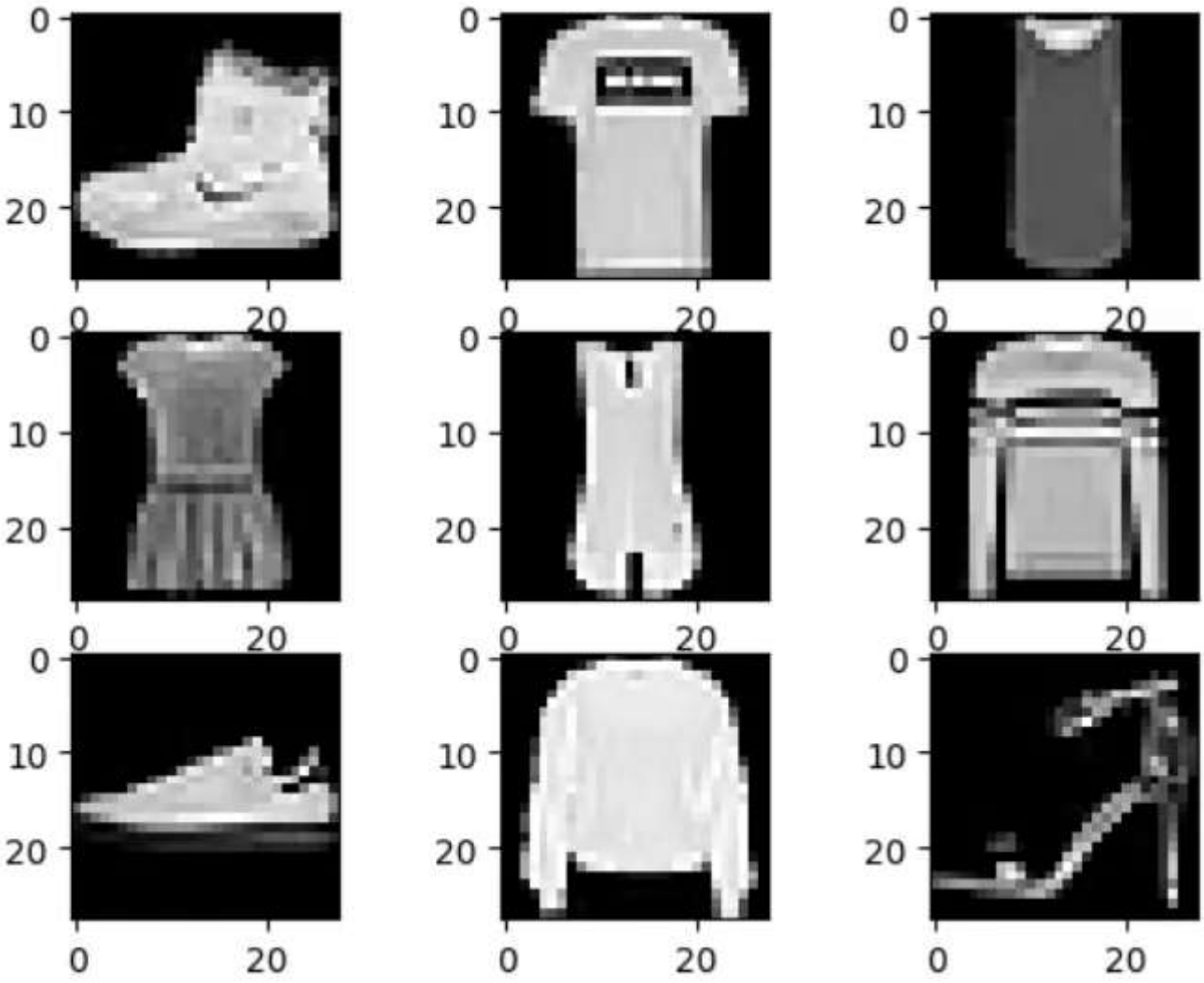
M Mayra Sarahí de Luna Castillo
11 min read · Nov 23

La clasificación de imágenes es una tarea dentro del campo de la visión por computadora que implica asignar una etiqueta o categoría a una imagen basándose en su contenido visual. Se trata de entrenar un modelo de aprendizaje automático para que pueda reconocer y asignar automáticamente una etiqueta a una imagen en función de patrones y características aprendidas durante el entrenamiento. En esta actividad, para comparar los diferentes modelos usados, se ha desarrollado una clasificación de imagen de tres bases de datos distintas con los algoritmos de SVM lineal , SVM con base radial y un modelo perceptrón multicapa.

Descripción de los datos

Para realizar este proyecto manejamos tres bases de datos. La primera es la Fashion-MNIST con 70000 imágenes escaladas y normalizadas (60000 de

entrenamiento y 10000 de prueba) de 10 clases de prendas de vestir (blusa, pantalón, suéter, vestido, abrigo, zapatos, playera, tenis, bolsa, botas) en escala de grises de tamaño 28x28. Las imágenes están perfectamente centradas y las distintas prendas tienen la misma figura.



Ejemplo de la base de datos Fashion-MNIST

La segunda es una base de 2000 fotografías satelitales de ambientes de México. Las fotografías son de alta calidad y muestran imágenes de Agua, Bosque, Ciudad, Cultivo, Desierto y Montaña.

Ejemplo de la base de datos fotografías satelitales

Por último, para la tercera base de datos, se utilizó una base de datos de 2,500 imágenes tomadas por nosotros, se tomaron 500 imágenes(con diferentes ángulos e iluminación) de cada una estas imágenes son de un zapato, una gorra, un celular, un lápiz y un libro.

Ejemplo de la base de datos de fotografías propias

Descripción de la teoría de los modelos entrenados y evaluados.

- **SVM Lineal (Support Vector Machine)**

SVM es un algoritmo de aprendizaje supervisado muy usado para modelos de clasificación. Este se centra en descubrir un hiperplano que logre separar de manera óptima dos clases en el espacio de características. El SVM lineal tiene como objetivo maximizar la distancia entre el hiperplano y los puntos más cercanos de ambas clases. La función objetivo del SVM es el siguiente

$$\min \frac{1}{2}\|w\|^2$$

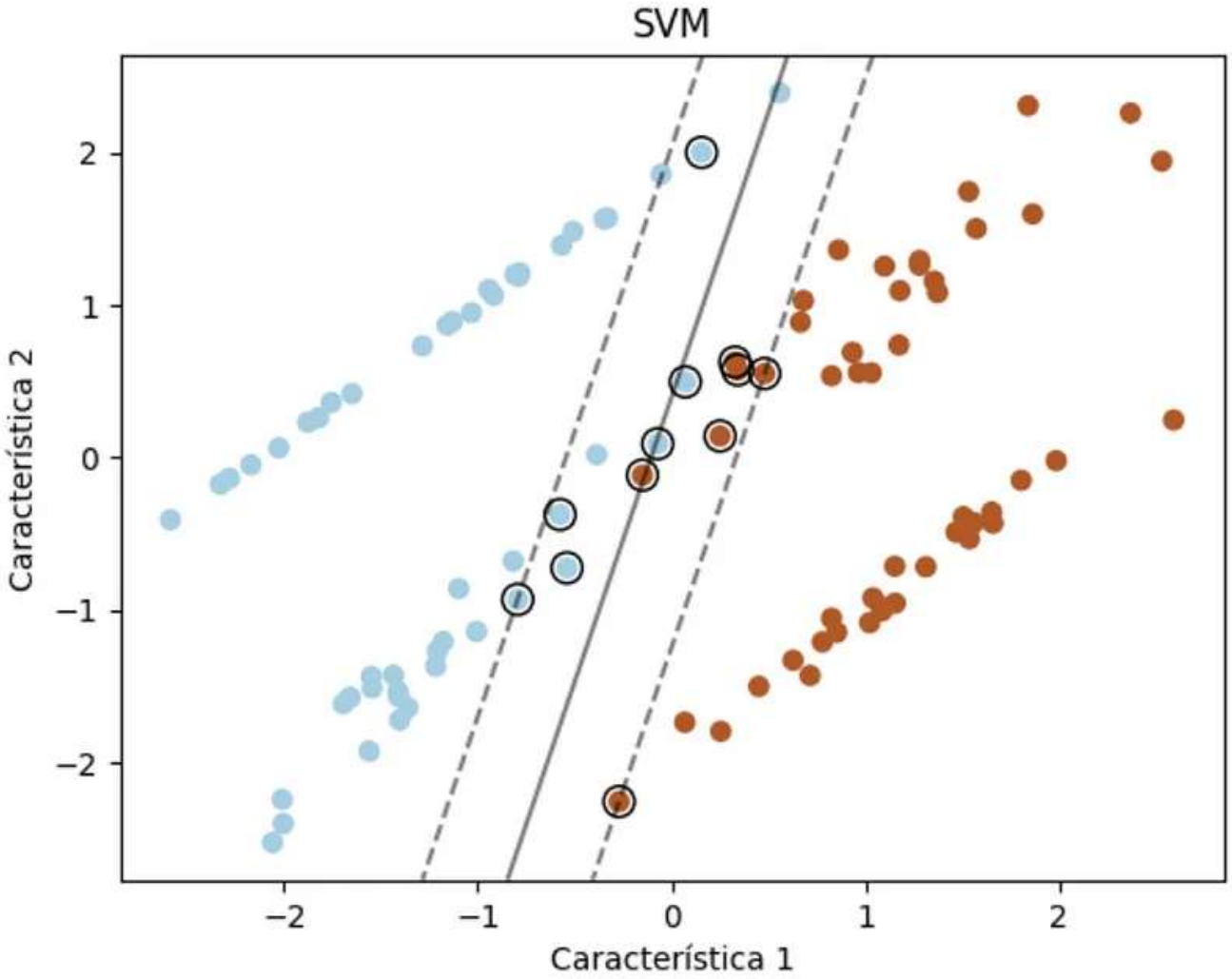
Función objetivo SVM

Sujeto a las restricciones:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, \quad i = 1, 2, \ldots, N.$$

Restricciones SVM Lineal

Para resolver este problema de optimización cuadrática, se utilizan algoritmos especializados en programación cuadrática, como Sequential Minimal Optimization (SMO) o métodos de descenso de gradiente.



Ejemplo SVM

- **SVM de base radial**

El kernel de base radial (RBF) es un tipo de función kernel utilizado en SVM para mapear los datos a un espacio de características de mayor dimensión. La función de base radial utiliza una medida de la distancia entre un punto de datos y un punto de referencia (o centroide), transformando la información en una representación más compleja y permitiendo la separación de clases no lineales.

En términos sencillos, el kernel RBF asigna pesos a los puntos de datos cercanos alrededor de un punto central, asignándoles valores más altos, y

disminuye estos pesos a medida que nos alejamos. Esto crea un modelo más flexible capaz de capturar patrones no lineales en los datos. La elección adecuada de los parámetros, como la anchura del kernel, es esencial para obtener un rendimiento óptimo en la clasificación de datos complejos.

$$K(x, x') = \exp(-\gamma \cdot ||x - x'||^2)$$

Función objetivo SVM base radial

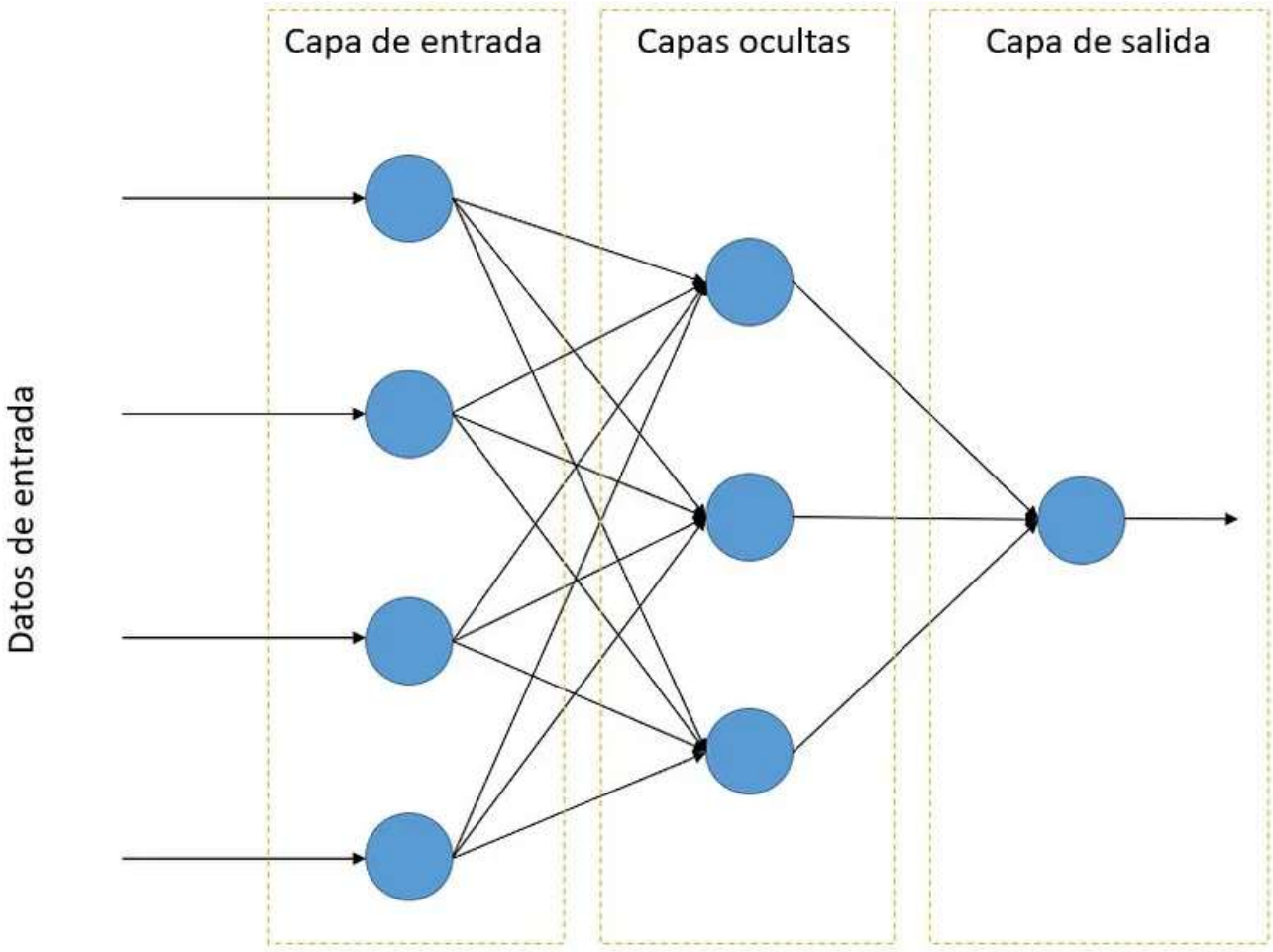
El proceso de entrenamiento y predicción para SVM con base radial implica encontrar los multiplicadores de Lagrange y el sesgo que satisfacen las restricciones y maximizan el margen en el espacio de características de mayor dimensión definido por la función de kernel radial. (Núcleo (Kernel) de las SVM, s.f.)

- **Perceptrón multicapa**

Es un tipo de red neuronal artificial organizada en capas. Consiste en al menos tres capas de nodos o unidades: una capa de entrada, una o varias capas ocultas y una capa de salida.

Cada nodo en una capa está conectado a todos los nodos de la capa siguiente y anterior mediante conexiones con pesos asociados. Estas conexiones transmiten información en forma de señales ponderadas. Cada nodo en una capa oculta realiza una operación lineal ponderada de las entradas, seguida de una función de activación no lineal, las principales funciones de activación incluyen ReLU (Unidad Lineal Rectificada), Sigmoide y Softplus. Cada una de estas funciones desempeña un papel fundamental en la

generación de la transformación deseada en los datos a medida que atraviesan las capas del modelo. Esto permite que la red pueda aprender relaciones no lineales en los datos. Durante el entrenamiento, la red ajusta los pesos de las conexiones para minimizar la diferencia entre las salidas predichas y las salidas reales, utilizando un algoritmo de optimización y una función de pérdida. (Perceptrón Multicapa, 2023)



El perceptrón multicapa. Recuperado de https://interactivechaos.com/es/manual/tutorial-de-deep-learning/el-perceptron-multicapa

Código en Python

A continuación se mostrarán los códigos empleados para cada modelo en cada base de datos:

- **Fashion-MNIST**

Se importó la base de datos Fashion-MNIST y se separa en X_train y X_test

```
from keras.datasets import fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
```

Se realizó el código para implementar un SVM lineal y calcular su accuracy y su recall por clase, en este caso se usa un SVC con kernel “linear”

```
x_train_flat = x_train.reshape(x_train.shape[0], -1)
x_test_flat = x_test.reshape(x_test.shape[0], -1)

linear_svm = SVC(kernel='linear')
linear_svm.fit(x_train_flat, y_train)

y_pred_svm_linear = linear_svm.predict(x_test_flat)

accuracy_linear = accuracy_score(y_test, y_pred_svm_linear)
recall_per_class = recall_score(y_test, y_pred_svm_linear, average=None)

print("Accuracy del modelo lineal SVM:", accuracy_linear)
for class_idx, recall in enumerate(recall_per_class):
    print(f"Recall para la clase {class_idx}: {recall}")
```

Se realizó el código que implementó un SVM con base radual para Fashion-MNIST y calcula su accuracy y recall por clase, en este caso se usa un SVC con kernel “rbf”.

```
rbf_svm = SVC(kernel='rbf')
rbf_svm.fit(x_train_flat, y_train)
```

```
y_pred_svm_rbf = rbf_svm.predict(x_test_flat)

accuracy_rbf = accuracy_score(y_test, y_pred_svm_rbf)
recall_per_class = recall_score(y_test, y_pred_svm_rbf, average=None)

print("Accuracy del modelo no lineal:", accuracy_rbf)
for class_idx, recall in enumerate(recall_per_class):
    print(f"Recall para la clase {class_idx}: {recall}")
```

Se realizó el código que implementó un perceptrón multicapa calculando su accuracy y recall por clase, además arroja la mejor configuración de capas y neuronas.

```
hidden_layer_sizes_options = [(50,), (100,), (50, 50), (100, 50), (100, 100)]

best_accuracy = 0
best_config = None

for hidden_layer_config in hidden_layer_sizes_options:
    mlp = MLPClassifier(hidden_layer_sizes=hidden_layer_config, max_iter=200, ra
    mlp.fit(x_train_flat, y_train)

    y_pred_test = mlp.predict(x_test_flat)
    accuracy = accuracy_score(y_test, y_pred_test)

    print(f"Configuración {hidden_layer_config}: Accuracy en test = {accuracy}")

    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_config = hidden_layer_config

final_mlp = MLPClassifier(hidden_layer_sizes=best_config, max_iter=200, random_s
final_mlp.fit(x_train_flat, y_train)

y_pred_test_final = final_mlp.predict(x_test_flat)
accuracy_final = accuracy_score(y_test, y_pred_test_final)
recall_final = recall_score(y_test, y_pred_test_final, average=None)

print("Mejor configuración de capas y neuronas:", best_config)
print("Accuracy:", accuracy_final)
```

```
for class_idx, recall in enumerate(recall_final):
    print(f"Recall para la clase {class_idx}: {recall}")
```

- **Fotografías satelitales**

Se re-escalaron los datos y las almacena en las listas “images” y “labels”, finalmente las convierte en un arreglo para su manipulación.

```
img_width = int(1920/scale)
img_height = int(1080/scale)

paths = ['']

images = []
labels = []
for label, p in enumerate(paths):
    files = [f for f in listdir(p) if isfile(join(p, f))]
    for f in files:
        print("Loading:", p + f)
        labels.append(label)
        rgb = io.imread(p + f)
        rgb_resized = resize(rgb, (img_height, img_width), anti_aliasing=True)
        images.append(rgb_resized)


n_img = len(images)
labels = np.array(labels)
```

Se extraen dos características de las imágenes: histogramas de color (frecuencias de color rojo, verde y azul), y descriptor de textura.

```
#   Histogramas de color
hist_descriptor = []
for rgb_resized in images:
  rh = np.histogram(rgb_resized[:,:,0].flatten(), nbins, density = True)
  gh = np.histogram(rgb_resized[:,:,1].flatten(), nbins, density = True)
  bh = np.histogram(rgb_resized[:,:,2].flatten(), nbins, density = True)
  hist_descriptor.append(np.concatenate((rh[0], gh[0], bh[0])))

#   Descriptor de textura
for rgb_resized in images:
  gray_resized = img_as_ubyte(rgb2gray(rgb_resized))
  glcm = graycomatrix(gray_resized, distances=[5], angles=[0, np.pi/4, np.pi/2,

  texture_desc.append([graycoprops(glcm, 'dissimilarity')[0, 0], graycoprops(glc
```

Se implementó el código para un SVM lineal, en este caso se utilizó validación cruzada de 5 divisiones debido a la cantidad de imágenes en este base (2,000), y se imprime la accuracy y el recall por clase.

```
x = np.concatenate((texture_desc, hist_descriptor), axis=1)

linear_svm = SVC(kernel='linear')

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

y_pred_svm_linear = cross_val_predict(linear_svm, x, labels, cv=cv)

recall_per_class = recall_score(labels, y_pred_svm_linear, average=None)

accuracy = accuracy_score(labels, y_pred_svm_linear)

for class_idx, recall in enumerate(recall_per_class):
    print(f"Recall para la clase {class_idx}: {recall}")

print("Accuracy:", accuracy)
```

Se implementó el código de SVM con base radial con un SVC con kernel “rbf” y validación cruzada de 5 divisiones, al igual que los anteriores, el código también imprime el accuracy y recall por clase.

```
linear_svm = SVC(kernel='rbf')

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

y_pred_svm_linear = cross_val_predict(linear_svm, x, labels, cv=cv)

recall_per_class = recall_score(labels, y_pred_svm_linear, average=None)

accuracy = accuracy_score(labels, y_pred_svm_linear)

for class_idx, recall in enumerate(recall_per_class):
    print(f"Recall para la clase {class_idx}: {recall}")

print("Accuracy:", accuracy)
```

Finalmente se implementó el código para un perceptrón multicapa con validación cruzada de 5 divisiones calculando su accuracy y recall por clase, además arroja la mejor configuración de capas y neuronas.

```
hidden_layer_sizes_options = [(50,), (100,), (50, 50), (100, 50), (100, 100)]

best_accuracy = 0
best_config = None

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

for hidden_layer_config in hidden_layer_sizes_options:
    mlp = MLPClassifier(hidden_layer_sizes=hidden_layer_config, max_iter=200, ra

    y_pred_cv = cross_val_predict(mlp, x, labels, cv=cv)

    accuracy = accuracy_score(labels, y_pred_cv)
```

```
        print(f"Configuración {hidden_layer_config}: Accuracy promedio en validación

        if accuracy > best_accuracy:
            best_accuracy = accuracy
            best_config = hidden_layer_config

final_mlp = MLPClassifier(hidden_layer_sizes=best_config, max_iter=200, random_s
final_mlp.fit(x, labels)

y_pred_final = final_mlp.predict(x)

recall_final = recall_score(labels, y_pred_final, average=None)

print("Mejor configuración de capas y neuronas:", best_config)
print("Accuracy promedio en validación cruzada:", best_accuracy)
for class_idx, recall in enumerate(recall_final):
    print(f"Recall para la clase {class_idx}: {recall}")
```

Por último, se realizaron los tres modelos para la base de datos de imágenes propias, los códigos son muy similares al de la base de datos de imágenes satelitales (es por ello que no se muestran) ya que también extrae las características de color y textura y usa validación cruzada por la cantidad de imágenes disponibles (2500).

Descripción de la metodología para evaluar los modelos seleccionados

Para evaluar los modelos realizados, se utilizaron dos métricas: Accuracy (precisión) y Recall (sensibilidad).

- Accuracy: Se define como el número de instancias de una clase específica que fueron correctamente clasificadas por el modelo dividido por el número total de instancias que realmente pertenecen a esa clase.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Fórmula para obtener la precisión

Esta métrica proporciona una medida general de la capacidad de un modelo para clasificar correctamente las instancias, sin embargo, cuando las clases están desequilibradas puede no ser una buena métrica, por eso es importante siempre complementar el análisis de la evaluación de lo de los modelos con otras métricas como recall, F1 o la especificidad.

- Recall: Se define como el número de instancias de una clase específica que fueron correctamente clasificadas por el modelo dividido por el número total de instancias que realmente pertenecen a esa clase.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Fórmula para Recall

El recall mide la capacidad del modelo para capturar o “recordar” todas las instancias positivas. Un valor alto de recall indica que el modelo es capaz de identificar la mayoría de las instancias positivas. (Koehrsen, 2023)

- Cross-Validation: Es una técnica utilizada en estadísticas y aprendizaje automático para evaluar el rendimiento de un modelo predictivo. El

> propósito de la validación cruzada es estimar cómo se comportará un modelo en un conjunto de datos independiente al que se utilizó para entrenarlo.

Para esto se divide el conjunto de datos disponible en dos partes: una para entrenar el modelo y otra para evaluar su rendimiento. La validación cruzada la realiza múltiples veces, de manera que se entrena y evalúa el modelo en diferentes subconjuntos de datos. Esto proporciona una evaluación más robusta del rendimiento del modelo, ya que se promedian los resultados obtenidos en las diferentes particiones. (Flores, 2022)

Resultados obtenidos con los diferentes modelos y conjuntos de datos

A continuación se muestran las tablas de los resultados de las métricas de comparación (accuracy y recall) con los tres distintos modelos implementados (SVM Lineal, SVM No Lineal, MLP) a las tres bases de datos.

TABLA DE RESULTADOS FASHION-MNIST			
	SVM Lineal	SVM No Lineal	MLP
Accuracy	0.846	0.882	0.884
Recall clase 0 (T-shirt/Top)	0.815	0.856	0.835
Recall clase 1 (Trouser)	0.962	0.962	0.968
Recall clase 2 (Pullover)	0.769	0.817	0.843
Recall clase 3 (Dress)	0.842	0.891	0.890
Recall clase 4 (Coat)	0.773	0.815	0.818
Recall clase 5 (Sandal)	0.936	0.951	0.951
Recall clase 6 (Shirt)	0.562	0.655	0.666
Recall clase 7 (Sneaker)	0.934	0.955	0.967
Recall clase 8 (Bag)	0.925	0.977	0.96
Recall clase 9 (Ankle boot)	0.945	0.951	0.951
Recall Promedio	0.7544	0.8739	0.8708

Tabla de resultados para la base de Fashion-MNSIT

TABLA DE RESULTADOS FOTOGRAFÍAS SATELITALES			
	SVM Lineal	SVM No Lineal	MLP
Accuracy	0.856	0.831	0.884
Recall clase 0 (Agua)	0.865	0.826	1
Recall clase 1 (Bosque)	0.844	0.818	0.996
Recall clase 2 (Ciudad)	0.937	0.904	0.997
Recall clase 3 (Cultivo)	0.723	0.611	0.996
Recall clase 4 (Desierto)	0.901	0.934	0.997
Recall clase 5 (Montaña)	0.839	0.847	0.997
Recall Promedio	0.851	0.824	0.997

Tabla de resultados para la base de datos de fotografías satelitales

TABLA DE RESULTADOS FOTOGRAFÍAS PROPIAS			
	SVM Lineal	SVM No Lineal	MLP
Accuracy	0.995	0.987	0.998
Recall clase 0 (Celulares)	0.984	0.958	1
Recall clase 1 (Cachuchas)	0.994	0.982	1
Recall clase 2 (Lápices)	1.0	1.0	1
Recall clase 3 (Libros)	1.0	1.0	1
Recall clase 4 (Tenis)	0.998	0.998	1
Recall Promedio	0.995	0.9876	1

Tabla de resultados para la base de datos de fotografías propias

Comparación de los resultados

Podemos obsevar que para la tabla de resultados del Fashion-MNIST , hubo dos modelos con una precisión buena, el mejor modelo en promedio fue el perceptrón multicapa (0.884 de accuracy) seguido del SVM no Lineal (0.882), tomando en cuenta también los recall de cada clase podemos obsevar que tanto en el SVM como en el perceptrón multicapa los resultados son muy parecidos y arrojan buenas métricas.

Para la base de datos de fotografías satelitales el mejor modelo al igual que en el ejemplo anterior fue el perceptrón multicapa (0.884 de accuracy) seguido del SVM Lineal (0.856 de accuracy), en este caso la diferencia de precisión si es notable, esto se puede deber a que las imágenes satelitales son un poco más complejas que las de la base de datos Fashion-MNIST, sin embargo el perceptrón multicapa tuvo una sensibilidad bastante buena, ya

que había clases en donde arrojó un 0.99 de clasificación de instancias positivas.

Finalmente, observamos los resultados arrojados para la base de datos de fotografías propias, en este caso, todas las precisiones son bastante buenas, arriba de 0.95, esto se debe a que el fondo de las imágenes es blanco, las fotografías son simples y en distintos ángulos, por lo que es muy fácil para los tres modelos clasificar de manera adecuada esta base. Además es importante analizar el recall del perceptrón multicapa, ya que en varias clases arroja una sensibilidad de 1, esto tiene sentido ya que MLP está diseñado para imágenes más complejas, por lo que es posible que para imágenes simples como estas, su clasificación sea perfecta.

Conclusiones

Como conclusión, y analizando los resultados obtenidos para los tres modelos de las tres bases de datos, se puede decir que tanto SVM lineal, como SVM de base radial y MLP fueron buenos modelos de clasificación, ya que en su precisión ninguno fue menor de 0.80, lo que nos indica que podríamos usar cualquiera de ellos y obtendríamos buenos resultados.

Además, es importante mencionar que aunque los tres modelos fueron buenos, el que arrojó siempre un mejor resultado fue el MLP (perceptrón multicapa) debido a que tienen la capacidad de aprender representaciones no lineales de los datos, tienen la capacidad de aprender automáticamente características relevantes durante el proceso de entrenamiento, y en general tienden a tener mejor desempeño en grandes conjuntos de datos, así que en este caso, el mejor modelo para nuestras bases de datos fue el MLP.

Referencias

Koehrsen, W. (2023). Precision and Recall : How to evaluate your classification model. *Built in*. Recuperado de https://builtin.com/data-science/precision-and-recall

Flores, N. (2022). Cross Validation: qué es y su relación con machine learning. *Tec de Monterrey*. Recuperado de https://blog.maestriasydiplomados.tec.mx/cross-validation-que-es-y-su-relacion-con-machine-learning

Núcleo (Kernel) de las SVM. (s.f.).*Numerentur*. Recuperado de https://numerentur.org/nucleo-kernel-de-las-svm/

Perceptrón Multicapa. (2023). *IBM*. Recuperado de https://www.ibm.com/docs/es/spss-statistics/saas?topic=networks-multilayer-perceptron

El Perceptrón Multicapa. (s.f.). InteractiveChaos. Recuperado de https://interactivechaos.com/es/manual/tutorial-de-deep-learning/el-perceptron-multicapa